TECHNICAL REPORT RE-75-32

A FORTRAN PROGRAM FOR THE CALCULATION OF THE STATE TRANSITION MATRIX AS A LINEAR COMBINATION OF REAL TIME FUNCTIONS (EAT)

Lewis G. Minor and John R. Underwood
Advanced Sensors Laboratory
US Army Missile Research, Development and Engineering Laboratory
US Army Missile Command
Redstone Arsenal, Alabama 35809

16 May 1975

Approved for public release; distribution unlimited.

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama

DISPOSITION INSTRUCTIONS

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT RETURN IT TO THE ORIGINATOR.

DISCLAIMER

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL INDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: (14) RE-75-32	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): (6) A FORTRAN PROGRAM FOR THE CALCULATION OF THE STATE TRANSITION MATRIX AS A LINEAR COMBINATION OF REAL TIME FUNCTIONS (EAT).		5. TYPE OF REPORT & PERIOD COVERED: (9) Technical Rep[redacted]t.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): (10) Lewis G. Minor [redacted] John R. Underwood		8. CONTRACT OR GRANT NUMBER(s): (12) 46p.
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Commander US Army Missile Command ATTN: AMSMI-RE Redstone Arsenal, Alabama 35809		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (DA) 1M362303A214 AMCMSC 632303.11.21401
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE: (11) 16 May [redacted]75
		13. NUMBER OF PAGES: 45
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): (16) DA-1-M-362303-A-214		15. SECURITY CLASS. (of this report): Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Cleared for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

State
Transition
Matrix
FORTRAN program
Linear differential equations
State variables

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Many programs are available which calculate numerically the state transition matrix at a specific time. However, programs which give the solution as a linear combination of real time functions are not generally available. The program given in this report calculates the state transition matrix as a linear combination of real time functions starting from any real system matrix.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 1103 086

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

CONTENTS

1. Introduction

The authors have developed a FORTRAN program that can be used to calculate the solution to the homogeneous system of first order differential equations with constant coefficients as follows:*

$$
\begin{aligned}
\dot{\bar{Y}}(t) &= \underline{A}\bar{Y}(t) \\
\bar{Y}(t_0) &= \bar{Y}_0 \quad .
\end{aligned}
\tag{1}
$$

Consider Equation (1) and let $\bar{Y}(t)$ be an n-vector of differentiable functions, $\bar{Y}_0$ be an n-vector of constants (initial conditions), and $\underline{A}$ be an $n \times n$ matrix with constant elements. Then the solution to Equation (1) can be written as

$$\bar{Y}(t) = \underline{\Phi}(t - t_0)\bar{Y}_0 \tag{2}$$

where $\underline{\Phi}$ is an $n \times n$ matrix known as the state transition matrix (fundamental matrix) whose entries are functions of t. For example,

$$
\begin{aligned}
\dot{y}_1(t) &= y_2(t) \\
\dot{y}_2(t) &= -2y_1(t) - 2y_2(t)
\end{aligned}
\tag{3}
$$

is a 2×2 system of linear first order equations, and $\underline{\Phi}(t)$ is the matrix

$$\begin{pmatrix} e^{-t}(\cos t - \sin t) & e^{-t} \sin t \\ -2\, e^{-t} \sin t & e^{-t}(\cos t - \sin t) \end{pmatrix} . \tag{4}$$

*The symbol $\underline{A}$ will be used to indicate A is a matrix and the symbol $\bar{A}$ will be used to indicate A is a column vector.

To determine a solution to the system, given a set of initial conditions,

$$\bar{Y}(t_0) = \begin{pmatrix} y_1(t_0) \\ y_2(t_0) \end{pmatrix} = \bar{Y}_0 = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} , \qquad (5)$$

one need only premultiply it by $\underline{\Phi}(t - t_0)$. This result has been in the literature for quite sometime[1] and is a special case of the more general result that

$$\begin{aligned} \dot{\bar{Y}}(t) &= \underline{A}\bar{Y}(t) + \underline{B}\bar{V}(t) \\ \bar{Y}(t_0) &= \bar{Y}_0 \end{aligned} \qquad (6)$$

is solved by

$$\underline{\Phi}(t - t_0)\bar{Y}_0 + \int_{t_0}^{t} \underline{\Phi}(t - t_0)\underline{B}\bar{V}(t)\,dt \quad . \qquad (7)$$

Clearly, the central problem in determining a particular solution in either the homogeneous or the more general problem is the calculation of $\underline{\Phi}(t)$, better known as $e^{\underline{A}t}$. Subroutines for calculating $e^{\underline{A}t}$ have been in the literature for quite sometime. All those known to the authors, however, have the drawback that they do not output $e^{\underline{A}t}$ in a form similar to that of Equation (4), but give $e^{\underline{A}t_a}$ for one value of $t = t_a$. This technique is widely used in the numerical integration of linear systems. Though such techniques are useful, the analytical form of the solution is lost together with time constants and system frequencies which appear in an analytical representation for $\underline{\Phi}(t)$.

The computer program developed in this report can be used to obtain $\underline{\Phi}(t)$ for every t. The user need only input the system matrix $\underline{A}$, the dimension of $\underline{A}$, and a set of error tolerance levels. For example, let

$$\underline{A} = \begin{pmatrix} 1 & 0 \\ -2 & -2 \end{pmatrix} \quad . \qquad (8)$$

---

[1] Frame, J. S., "Matrix Functions and Applications, IV," IEEE Spectrum, June 1964, pp. 123-131.

Then the program output will contain (among other things) $\underline{\Phi}(t)$ expressed as a sum of matrices times linearly independent functions:

$$\underline{\Phi}(t) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} e^{-t} \cos t + \begin{pmatrix} -1 & 1 \\ -2 & -1 \end{pmatrix} e^{-t} \sin t \quad . \tag{9}$$

The error tolerance levels are discussed in detail in Section 3.

The remainder of this report is divided into three sections. Section 2 gives a description of the problem and the techniques used to solve it. Section 3 deals exclusively with the computer program and includes a description of inputs and outputs. Finally, Section 4 gives a listing of the program. Those familiar with analytic functions of matrices can go directly to Section 3.

2. The State Transition Matrix

a. $e^{\underline{A}t}$, Definition, and Properties

$e^{\underline{A}t}$, the $\underline{\Phi}(t)$ of Section 1, is defined by a power series as

$$e^{\underline{A}t} = \underline{I} + \underline{A}t + \frac{\underline{A}^2 t^2}{2!} + \frac{\underline{A}^3 t^3}{3!} + \ldots \tag{10}$$

where $\underline{I}$ represents the unit matrix. Since $e^{\underline{A}t}$ is defined analogously to $e^{\alpha t}$, α is a scaler, it is similar in many respects. For example,

$$e^{\underline{A}(t+s)} = e^{\underline{A}t} e^{\underline{A}s}$$

$$\frac{d\, e^{\underline{A}t}}{dt} = \underline{A}\, e^{\underline{A}t}$$

$$e^{\underline{A}\cdot 0} = \underline{I} \quad ; \tag{11}$$

but, generally speaking

$$e^{(\underline{A}t + \underline{B}t)} \neq e^{\underline{A}t} e^{\underline{B}t}$$

$$e^{\underline{A}t} e^{\underline{B}t} \neq e^{\underline{B}t} e^{\underline{A}t} \tag{12}$$

since matrix multiplication is not a commutative operation. A computationally useful property of $e^{\underline{A}t}$ is

$$e^{\underline{RAR}^{-1}} = \underline{R}e^{\underline{A}t}\underline{R}^{-1} . \tag{13}$$

This makes it possible to easily calculate $e^{\underline{A}t}$ in some cases by transforming it into a similar matrix (e.g., normal matrices to diagonal matrices[2]). The power series representation, while useful for numerical work, is none too helpful for writing $e^{\underline{A}t}$ in closed form. The following general theorem is used to decompose $e^{\underline{A}t}$ into a finite sum of $n \times n$ matrices times analytic functions in one variable.

Theorem[3]

If f is an analytic function on a simply connected open set D of the complex plane that contains all the eigenvalues λ_j of an $n \times n$ matrix $\underline{B}$ and the origin, then $f(\underline{B})$ may be written as

$$f(\underline{B}) = \sum_{j=1}^{s} \sum_{k=1}^{n_j} \frac{f^{(k-1)}(\lambda_j)}{(k-1)!} \underline{Z}_{j,k} \tag{14}$$

where n_j denotes the multiplicity of the j^{th} eigenvalue, s is the number of eigenvalues, and the $\underline{Z}_{j,k}$ are $n \times n$ matrices which are independent of f and D (they depend only on the matrix $\underline{B}$).

Since e^x is analytic in the whole complex plane, setting $\underline{B} = \underline{A}t$, $f(x) = e^x$ yields

$$e^{\underline{A}t} = \sum_{j=1}^{s} \sum_{k=1}^{n_j} \frac{t^{k-1} e^{\lambda_j t}}{(k-1)!} \underline{Z}_{j,k} . \tag{15}$$

Now it is clear that $e^{\underline{A}t}$ has a representation as a finite sum of $n \times n$ matrices of scalers times analytic functions. The problem is to determine the λ_j and the $\underline{Z}_{j,k}$.

---

[2] Herstein, I. N., Topics in Algebra, Blaisdell, Waltham, Massachusetts, 1964.

[3] Frame, loc. cit.

A review of the terminology involved in the theorem is presented. A function analytic about {0} has a power series representation about {0}:

$$f(z) = \sum_{n=0}^{\infty} a_n z^n \text{ for } |z| < r \quad ; \tag{16}$$

therefore, $f(\underline{B})$ can be formally defined by

$$f(\underline{B}) = \sum_{n=0}^{\infty} a_n \underline{B}^n \quad . \tag{17}$$

If $\underline{B}$ is an $n \times n$ matrix, an eigenvalue of $\underline{B}$ is a scaler λ_j for which a vector $\bar{x}$ exists such that

$$\underline{B}\bar{x} = \lambda_j \bar{x} \tag{18}$$

or

$$(\underline{B} - \lambda_j \underline{I})\bar{x} = \underline{0} \quad . \tag{19}$$

If Equation (19) holds, the matrix $\underline{B} - \lambda \underline{I}$ is singular, and, therefore, λ_j is a solution of the equation,

$$\text{Det}(\underline{B} - \lambda \underline{I}) = 0 \quad , \tag{20}$$

where $\text{Det}(\underline{B} - \lambda \underline{I})$ is an n^{th} degree polynomial called the characteristic polynomial of $\underline{B}$. Its roots, which are just the eigenvalues of $\underline{B}$, are also called the characteristic roots of $\underline{B}$. The multiplicity of λ_j is its multiplicity as a root of $\text{Det}(\underline{B} - \lambda \underline{I})$.

The theorem asserts that $f(\underline{B})$ exists (the power series of Equation (17) converges) if the eigenvalues are in D and that $f(\underline{B})$ has a representation in the form of Equation (14).

b. <u>The Characteristic Polynomial and Its Roots</u>

Certainly, the first problem in writing $e^{\underline{A}t}$ in the form of Equation (15) is to calculate the eigenvalues and determine their multiplicities. To do this the characteristic polynomial has to be calculated. If

$$\text{Det}(\underline{A} - \lambda \underline{I}) = \lambda^n + d_1 \lambda^{n-1} + \ldots + d_0 \tag{21}$$

is the characteristic polynomial, the coefficients d_k are calculated using the following theorem.

Theorem[4]

$$d_k = -\mathrm{tr}(\underline{AB}_{k-1})$$

$$\underline{B}_k = \underline{AB}_{k-1} + d_k \underline{I}$$

$$d_0 = 1, \underline{B}_0 = I, \underline{B}_n = \underline{0}$$

$$0 \leq k \leq n \quad . \tag{22}$$

Recall that $\mathrm{tr}(a_{ij}) = \Sigma a_{ii}$ is the sum of the diagonal entries of (a_{ij}). Since $\underline{B}_n = \underline{0}$ a good way to check for errors in the calculation of the coefficients is to check the difference between the calculated $\underline{B}_n$ and the theoretically determined values. An interesting consequence of Equation (21) is that

$$\frac{\underline{AB}_{n-1}}{-d_n} = \underline{I} \tag{23}$$

or

$$\frac{\underline{B}_{n-1}}{-d_n} = A^{-1} \quad , \tag{24}$$

if $\underline{A}^{-1}$ exists $(d_n \neq 0)$.

The roots of $\mathrm{Det}(\underline{A} - \lambda\underline{I})$ can, at this stage, be calculated using any one of a number of different techniques. In this program the classical Newton-Raphson method is used. The roots then have to be sorted and multiplicities counted. Numerical errors can be generated almost anywhere in our program. At this stage, these errors necessitate a decision. For example, using Newton-Raphson the equation

$$x^4 + 2x^2 + 1 = 0 \tag{25}$$

will have calculated roots $\epsilon_1 \pm i$, $\epsilon_2 \pm i$ where ϵ_1, ϵ_2 are small ($< 10^{-13}$ using a double precision version of an IBM root routine), but distinct real numbers. The solution to a 4×4 system with Equation (25) as its characteristic equation will be calculated to be of the form

[4] Frame, *op. cit.*

$$\underline{Z}_{1,1}e^{(\epsilon_1+i)t} + \underline{Z}_{2,1}e^{(\epsilon_1-i)t} + \underline{Z}_{3,1}e^{(\epsilon_2+i)t} + \underline{Z}_{1,4}e^{(\epsilon_2-i)t} \quad . \qquad (26)$$

But since the roots of Equation (25) are really $\pm i$, each of multiplicity two, the solution really is

$$\underline{Z}_{1,1}e^{it} + \underline{Z}_{1,2}te^{it} + \underline{Z}_{2,1}e^{-it} + \underline{Z}_{2,2}te^{-it} \quad . \qquad (27)$$

To avoid this kind of problem some decision has to be made. Namely, a tolerance ϵ is specified such that a will be set equal to b if

$$|a - b| < \epsilon \qquad (28)$$

where a and b are roots of the characteristic polynomial.

For example, in the case previously considered, ϵ is set to be large enough so that

$$|\epsilon_1 - \epsilon_2| < \epsilon \quad . \qquad (29)$$

The program replaces ϵ_2 by ϵ_1 and, rather than saying that the characteristic polynomial has distinct roots $\epsilon_1 \pm i$ and $\epsilon_2 \pm i$, claims that it has a root of $\epsilon_1 \pm i$ of multiplicity two. The program then checks to determine if the root, or its real or imaginary part, is small enough to be called zero, i.e.,

$$|a| < \epsilon \quad . \qquad (30)$$

If Equation (30) is satisifed, a is set equal to zero. In the example, the final output is two roots, $\pm i$ (each of multiplicity two), if ϵ is sufficiently large.

Although an error at this stage will make the solution to the system look radically different, it will probably not change any of the usual system constants in a discontinuous manner. In fact, the solution to Equation (1) depends continuously on the eigenvalues. To see this, notice from Equation (10) that the solution depends continuously on $\underline{A}$. Using Equation (13) it may be assumed that $\underline{A}$ is in lower triangular form[5]

---

[5]Herstein, loc. cit.

$$\underline{A} = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ * & & \lambda_s \end{pmatrix} \quad . \tag{31}$$

The diagonal entries of $\underline{A}$ are just the eigenvalues. If the roots are changed by a small amount ϵ_i, then the system is changed to one with matrix $\underline{A}'$ where

$$\underline{A}' = \underline{A} + \underline{\epsilon} \tag{32}$$

$$\underline{\epsilon} = \begin{pmatrix} \epsilon_1 & & 0 \\ & \ddots & \\ 0 & & \epsilon_s \end{pmatrix} \quad .$$

Hence,

$$e^{\underline{A}'t} = e^{(\underline{A}+\epsilon)t} \tag{33}$$

which is continuous in the ϵ_i.

c. <u>The Constituent Matrices</u>

The matrices $\underline{Z}_{j,k}$ are called the constituent matrices and, as was previously noted, are dependent only on $\underline{A}$, not on the analytic function at which $\underline{A}$ is evaluated. Rather than launch into a general description of the technique used to calculate the constituent matrices, they will first be calculated for a particular example and the illustrated technique generalized. Suppose

$$\underline{A} = \begin{pmatrix} 0 & 1 & 3 \\ 6 & 0 & 2 \\ -5 & 2 & 4 \end{pmatrix} \tag{34}$$

with characteristic polynomial

$$(x - 1)^2(x - 2) \tag{35}$$

which has roots $\lambda_1 = 1$ (of multiplicity two) and $\lambda_2 = 2$ (with multiplicity one). $e^{\underline{A}t}$ must be of the form

$$e^{\underline{A}t} = e^t\underline{Z}_{1,1} + te^t\underline{Z}_{1,2} + e^{2t}\underline{Z}_{2,1} \quad . \tag{36}$$

Applying Theorem 1 to the analytic functions $f(x) = x^0$, $g(x) = x$, and $h(x) = x^2$ and substituting $\underline{A}$ for x, the resulting equations are

$$\underline{I} = 1 \cdot \underline{Z}_{1,1} + 0\ \underline{Z}_{1,2} + 1\ \underline{Z}_{2,1}$$

$$\underline{A} = 1 \cdot \underline{Z}_{1,1} + 1 \cdot \underline{Z}_{1,2} + 2 \cdot \underline{Z}_{2,1}$$

$$\underline{A}^2 = 1 \cdot \underline{Z}_{1,1} + 2 \cdot \underline{Z}_{1,2} + 4 \cdot \underline{Z}_{2,1} \quad . \tag{37}$$

By using matrix notation and considering $\underline{I}$, $\underline{A}$, $\underline{A}^2$, $\underline{Z}_{j,k}$ as formal symbols Equation (37) can be written as

$$\begin{pmatrix} \underline{I} \\ \underline{A} \\ \underline{A}^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 4 \end{pmatrix} \begin{pmatrix} \underline{Z}_{1,1} \\ \underline{Z}_{1,2} \\ \underline{Z}_{2,1} \end{pmatrix} . \tag{38}$$

The matrix

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 4 \end{pmatrix} \tag{39}$$

is invertible with inverse

$$\begin{pmatrix} 0 & 2 & -1 \\ -2 & 3 & -1 \\ 1 & -2 & 1 \end{pmatrix} \tag{40}$$

so

$$\begin{pmatrix} \underline{Z}_{1,1} \\ \underline{Z}_{1,2} \\ \underline{Z}_{2,1} \end{pmatrix} = \begin{pmatrix} 0 & 2 & -1 \\ -2 & 3 & -1 \\ 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} \underline{I} \\ \underline{A} \\ \underline{A}^2 \end{pmatrix} \tag{41}$$

or, in equation form,

$$\underline{Z}_{1,1} = 2\underline{A} - \underline{A}^2$$

$$\underline{Z}_{1,2} = -2\underline{I} + 3\underline{A} - \underline{A}^2$$

$$\underline{Z}_{2,1} = \underline{I} - 2\underline{A} + \underline{A}^2 \quad . \tag{42}$$

The $\underline{Z}$'s can be easily calculated from Equation (42). In general, if $\underline{A}$ is an $n \times n$ matrix, the column vectors

$$\bar{A}_a = \begin{pmatrix} \underline{I} \\ \underline{A} \\ \vdots \\ \underline{A}^{n-1} \end{pmatrix} \tag{43}$$

$$\bar{Z}_a = \begin{pmatrix} \underline{Z}_{1,1} \\ \underline{Z}_{1,2} \\ \vdots \\ \underline{Z}_{s,n_s} \end{pmatrix} \tag{44}$$

are formed and yield the matrix equation

$$\bar{A}_a = \underline{V}^t \bar{Z}_a \tag{45}$$

where $\underline{V}^t$ is the transpose of an $n \times n$ matrix $\underline{V}$, called the Vandermonde of Equation (1). The simplest way to calculate the entries of $\underline{V}^t$ is as follows. Partition $\underline{V}^t$ into s, $n \times n_s$ matrices. The k^{th} matrix will be filled with entries calculated from the k^{th} eigenvalue. The entries of each of these matrices is calculated using the algorithm

$$\begin{aligned} v_{ij} &= 0 && i > j \\ v_{ij} &= 1 && i = j \\ v_{ij} &= v_{i-1,j-1} + \lambda_k v_{i-1,j} && i < j \ . \end{aligned} \tag{46}$$

As an example, suppose $\underline{A}$ has 3 eigenvalues λ_1 of multiplicity 3, λ_2 of multiplicity 2, and λ_3 with multiplicity 1, then $\underline{V}^t$ is

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ \lambda_1 & 1 & 0 & \lambda_2 & 1 & \lambda_3 \\ \lambda_1^2 & 2\lambda_1 & 1 & \lambda_2^2 & 2\lambda_2 & \lambda_3^2 \\ \lambda_1^3 & 3\lambda_1^2 & 3\lambda_1 & \lambda_2^3 & 3\lambda_2^2 & \lambda_3^3 \\ \lambda_1^4 & 4\lambda_1^3 & 6\lambda_1^2 & \lambda_2^4 & 4\lambda_2^3 & \lambda_3^4 \\ \lambda_1^5 & 5\lambda_1^4 & 10\lambda_1^3 & \lambda_2^5 & 5\lambda_2^4 & \lambda_3^5 \end{pmatrix} \tag{47}$$

where $\underline{V}^t$ is always invertible[6] so Equation (45) can always be solved.

d. Complex Eigenvalues

In all previous examples the eigenvalues have been real. In general, the eigenvalues may be complex and the result is that λ_j and $\underline{Z}_{i,j}$ may in general be complex. The program developed in this report handles all complex computations with real computation and no FORTRAN complex declarations are made. As a result, double precision may be used to provide accurate solutions.

If the eigenvalues are complex then the matrix $e^{\underline{A}t}$ may contain numbers and functions which are complex in form and must be combined to form a solution which contains only real numbers and real functions. The task is tedious by hand, so the program combines complex functions into a real form convenient to the user.

The generation of a real form for $e^{\underline{A}t}$ with complex eigenvalues is handled in the same manner as in the case of a single linear equation with constant coefficients. Namely, if $\lambda_j = \alpha + i\beta$ is a characteristic root then so is $\lambda_\ell = \alpha - i\beta$, and the two roots have the same multiplicity. The term

$$\frac{t^{k-1}}{(k-1)!} \left(e^{\alpha t} e^{i\beta t} \underline{Z}_{j,k} + e^{\alpha t} e^{-i\beta t} \underline{Z}_{\ell,k} \right) \tag{48}$$

[6]Frame, loc. cit.

occurs in $e^{\underline{A}t}$. Write

$$\underline{Z}_{j,k} = \underline{Z} + i\underline{ZZ} \tag{49}$$

$$\underline{Z}_{\ell,k} = \underline{W} + i\underline{WW} \tag{50}$$

with $\underline{W}$, $\underline{WW}$, $\underline{Z}$, $\underline{ZZ}$ real $n \times n$ matrices, and write $e^{\pm i\beta t}$ as

$$\cos \beta t \pm i \sin \beta t \quad . \tag{51}$$

Multiplying out the complex numbers and grouping terms, Equation (48) becomes

$$\frac{t^{k-1} e^{\alpha t}}{(k - 1)!}\left[\cos \beta t(\underline{Z}+\underline{W}) + \sin \beta t(\underline{WW} - \underline{ZZ}) + i \left\{\sin \beta t(\underline{Z} - \underline{W}) + \cos \beta t (\underline{ZZ} + \underline{WW})\right\}\right] \tag{52}$$

but $e^{\underline{A}t}$ is real since $\underline{A}$ is, therefore, the complex part of Equation (52) must vanish for all t. By evaluating Equation (52) at 0 and π/β, the connections

$$\begin{aligned} \underline{Z} &= \underline{W} \\ \underline{ZZ} &= -\underline{WW} \end{aligned} \tag{53}$$

are established. Simplifying Equation (52) gives

$$\frac{t^{k-1}}{(k - 1)!} e^{\alpha t} \left[2 \cos \beta t \cdot \underline{Z} - 2 \sin \beta t \cdot \underline{ZZ}\right] \quad . \tag{54}$$

All the quantities in Equation (54) are real.

It will now be shown how the complex computations of Equation (45) may be handled using only real computations. Recall that complex numbers can be changed into 2×2 matrices according to the following rule:

$$\alpha + i\beta \longmapsto \begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix} \quad . \tag{55}$$

If z and w are complex and $\underline{Z}$, $\underline{W}$ are their corresponding matrices then[7,8]

---

[7] Frame, op. cit.

[8] Herstein, loc. cit.

$$
\begin{aligned}
z + w &\longmapsto \underline{Z} + \underline{W} \\
zw &\longmapsto \underline{ZW} \\
z^{-1} &\longmapsto \underline{Z}^{-1} \quad .
\end{aligned} \tag{56}
$$

Equations (56) are sufficient to guarantee that algebraic calculations done with the matrices will agree with those done with their complex numbers. This monomorphism is extended to n × n matrices in the natural way. If $\underline{Z}$ is a complex n × n matrix and

$$\underline{Z} = \underline{A} + i\underline{B} \quad , \tag{57}$$

then

$$\underline{Z} \longmapsto \begin{pmatrix} \underline{A} & -\underline{B} \\ \underline{B} & \underline{A} \end{pmatrix} \quad , \tag{58}$$

a 2n × 2n matrix.

Similarily, if $\underline{W}$, $\underline{C}$, and $\underline{D}$ are n × 1 matrices such that

$$\underline{W} = \underline{C} + i\underline{D} \quad , \tag{59}$$

then

$$W \longmapsto \begin{pmatrix} \underline{C} \\ \underline{D} \end{pmatrix} \quad . \tag{60}$$

Equations analogous to Equation (56) hold.[9,10] Under these transformations, Equation (59) becomes

---

[9] Frame, loc. cit.

[10] Bredon, G. E., Introduction to Compact Transformation Groups, Academic Press, New York, New York, 1972.

$$\begin{pmatrix} \underline{I} \\ \underline{A} \\ \vdots \\ \underline{A}^{n-1} \\ \underline{0} \\ \vdots \\ \underline{0} \end{pmatrix} = \begin{pmatrix} \underline{VR}^t & -\underline{VC}^t \\ \underline{VC}^t & \underline{VR}^t \end{pmatrix} \begin{pmatrix} \underline{ZR}_{1,1} \\ \vdots \\ \underline{ZR}_{n,n_s} \\ \underline{ZI}_{1,1} \\ \vdots \\ \underline{ZI}_{n,n_s} \end{pmatrix} \tag{61}$$

where $\underline{V}^t = \underline{VR}^t + i\underline{VC}^t$ and $\underline{Z}_{i,j} = \underline{ZR}_{i,j} + i\underline{ZI}_{i,j}$. The $2n \times 2n$ matrix

$$\left(\begin{array}{c|c} \underline{VR}^t & -\underline{VC}^t \\ \hline \underline{VC}^t & \underline{VR}^t \end{array}\right) \tag{62}$$

is inverted and the calculations are made as before.

The $2n \times 2n$ matrix given in Equation (62) cannot be inverted by inverting the two $n \times n$ matrices $\underline{VR}$ and $\underline{VC}$ as can be shown by this counter-example:

$$\underline{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \quad ; \tag{63}$$

hence

$$\text{Det}(\underline{A} - \lambda\underline{I}) = -(\lambda^3 + \lambda) \tag{64}$$

so

$$\lambda = 0, \pm i \quad . \tag{65}$$

$\underline{V}^t$ is

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & i & -i \\ 0 & -1 & -1 \end{pmatrix} \quad . \tag{66}$$

The real matrix corresponding to $\underline{V}^t$ is

$$
\left(\begin{array}{ccc|ccc}
1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 \\
0 & -1 & -1 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & -1
\end{array}\right) \tag{67}
$$

None of the 3×3 matrices are invertible.

e. Error Checks

Recall that $e^{\underline{A}t}$ satisfies

$$\frac{de^{\underline{A}t}}{dt} = \underline{A}e^{\underline{A}t} . \tag{68}$$

Furthermore, among all the analytic functions satisfying Equation (68) it alone satisfies

$$f(0) = \underline{I} . \tag{69}$$

Equations (68) and (69) can be used to construct a test on the validity of any technique purporting to calculate $e^{\underline{A}t}$ in terms of its constituent matrices. If $e^{\underline{A}t}$ is differentiated with respect to t [Equation (15)], Equation (70) results

$$\sum_{j=1}^{s} \sum_{k=1}^{n_j} \frac{t^{k-1} e^{\lambda_j t}}{(k-1)!} \left[\underline{Z}_{j,k+1} + \lambda_j \underline{Z}_{j,k}\right] \tag{70}$$

(without loss of generality we can set $\underline{Z}_{j}, n_{j}+1 = \underline{0}$). In terms of Equation (68)

$$\sum_{j=1}^{s} \sum_{k=1}^{n_j} \frac{t^{k-1} e^{\lambda_j t}}{(k-1)!} \left[\underline{Z}_{j,k+1} + \lambda_j \underline{Z}_{j,k} - \underline{A}\underline{Z}_{j,k}\right] = \underline{0} \tag{71}$$

or setting

$$\frac{1}{(k-1)!} \underline{Z}_{j,k} = \underline{Z}'_{j,k} \tag{72}$$

$$\sum_{j=1}^{s} \sum_{k=1}^{n_j} t^{k-1} e^{\lambda_j t} \left[k\underline{Z}_{j,k+1} + \lambda_j \underline{Z}_{j,k} - \underline{AZ}_{j,k} \right] = \underline{0} \quad . \tag{73}$$

The functions $t^{k-1} e^{\lambda_j t}$ are linearly independent so their coefficient must be zero for the left hand side of Equation (73) to equal the null matrix:

$$k\underline{Z}'_{j,k+1} + \lambda_j Z'_{j,k} - \underline{AZ}'_{j,k} = \underline{0} \quad k < n_j$$

$$\lambda_j \underline{Z}'_{j,n_j} - \underline{AZ}_{j,n_j} = \underline{0} \quad . \tag{74}$$

If $\lambda_j = \alpha + i\beta$ is complex then, using analogous notation,

$$k\underline{ZR}'_{j,k+1} + \alpha\underline{ZR}'_{j,k} + \beta\underline{ZI}'_{j,k} - \underline{AZR}'_{j,k} = \underline{0}$$

$$k\underline{ZI}'_{j,k+1} + \alpha\underline{ZI}'_{j,k} - \beta\underline{ZR}'_{j,k} - \underline{AZI}'_{j,k} = \underline{0}$$

$$\alpha\underline{ZR}'_{j,n_j} + \beta\underline{ZI}'_{j,n_j} - \underline{AZR}'_{j,n_j} = \underline{0}$$

$$\alpha\underline{ZI}'_{j,n_j} - \beta\underline{ZR}'_{j,n_j} - \underline{AZI}'_{j,n_j} = \underline{0} \quad . \tag{75}$$

Checking these equations provides a good overall check for numerical errors. The program keeps track of the maximum entry of the matrix on the left hand sides of Equations (74) and (75) (absolute value of the matrix entry). After all calculations have been done, it prints out this maximum.

3. Program Description

a. Flow Chart of Program

A flow chart of the program containing the computations defined in Section 2 is given in Figure 1. All blocks in the flow chart with the exception of the last, are either self-explanatory or have already been explained. In calculating the inverse of a matrix the IBM subroutine INVERT,[11] which employs pivotal condensation, was used. In

[11] System/360 Scientific Subroutine Package, International Business Machines, While Plains, New York, 1968.

calculating the roots of the characteristic polynomial POLRT, another IBM routine was used.[12] It is a 500-step Newton-Raphson method in two variables.

b. Inputs to Program

This section defines the form of the input data for the program. The first four cards define the tolerance constants described in Section 2. These cards are read only once for any set of $\underline{A}$ matrices to be run. If the dimension card (LDEM) for an $\underline{A}$ matrix is zero in value, the computer run will terminate.

The order and format of the input data cards are shown in Figure 2. The variable names in Figure 2 can be identified as follows:

DB - In sorting roots, two roots which differ by an amount less than DB are set equal. This tolerance will effect the multiplicity calculated for a root and hence the form of the solution. If the user selects DB = 0, distinct roots are likely to be generated changing the form of the solution. Since the solution to Equation (1) depends continuously on the eigenvalues, the form of the solution chosen by the program will give the correct answer in a numeric sense.

DD - If the determinant of the Vandermonde is less than DD in absolute value, an error message is printed stating that the Vandermonde matrix is singular. When this message is encountered, one of the following occurred: (1) DD was selected too large, (2) User entered data improperly, or (3) numerical roundoff or truncation error is the source of the problem. After the error message is printed, execution is halted and the next data set is read in (starting with LDEM). If the user sets DD = 0, the program halts only if the determinant of $\underline{V}$ is zero.

LDEM - Order of the system matrix.

A(I,J) - Element of the system matrix.

c. Sample Program Output

Along with the error messages already discussed, the program will print the $\underline{A}$ matrix, its characteristic polynomial, a table of characteristic roots and their multiplicities, $\underline{e}^{\underline{A}t}$ expressed as a sum of constituent matrices and analytic functions, and the maximum error as determined by the method of Section 2.e.

[12] Ibid.

The input data set shown in Figure 3 was used to generate the output shown in Figure 4.

4. Program Listing

The listing given in Figure 5 is the double precision version of the EAT program.

5. Conclusions

Programs to calculate the state transition matrix ($e^{\underline{A}t}$) as a linear combination of functions of time are not generally available. The program described in the report computes $e^{\underline{A}t}$ and should be useful to those involved in the solution of linear differential equations described by state space equations.

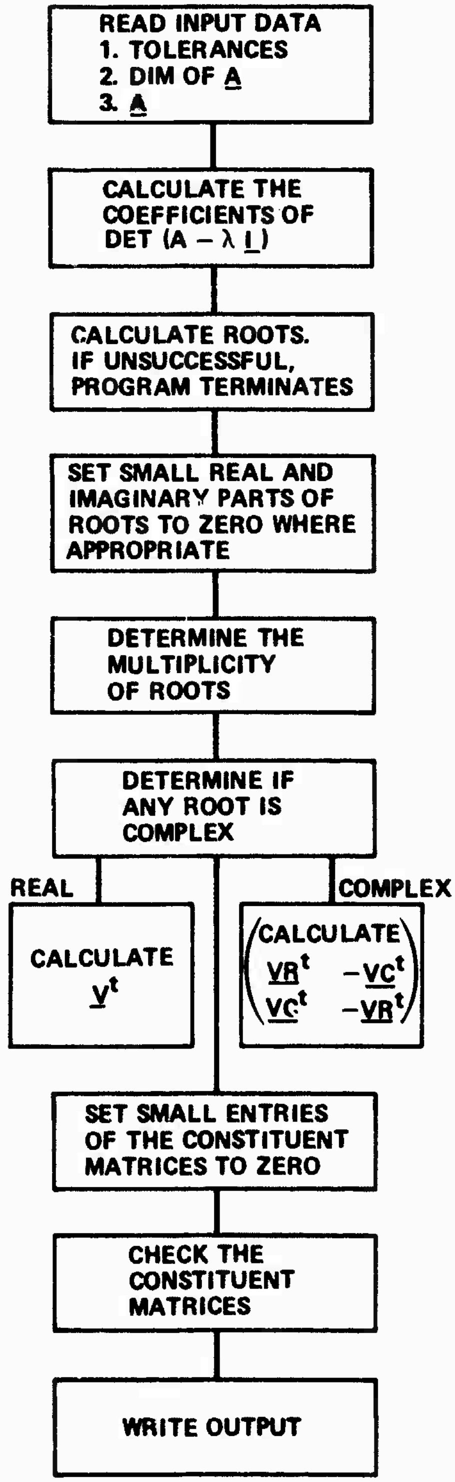

Figure 1. Flow chart of program.

	CARD NO.	VARIABLE NAME	FORMAT
TOLERANCE LEVELS	1	DB	D15.0
	2	DC	D15.0
	3	DD	D15.0
FIRST DATA SET	4	LDEM	I2
	5	A(1,1), A(1,2), A(1,3), A(1,4)	4D15.0
	.	.	.
	.	.	.
	.	.	.
SECOND DATA SET	M	LDEM	I2
	M+1	A(1,1), A(1,2), A(1,3), A(1,4)	4D15.0
	.	.	.
	.	.	.
	.	.	.
RUN TERMINATION	K	BLANK CARD	I2

Figure 2. Input data card format.

5	15	30	45	60 ← CARD COLUMN NUMBER
0.0001				
0.0001				
0.0001				
0.0001				
4				
0.0	1.0	0.0	0.0	1ST DATA SET
0.0	0.0	1.0	0.0	
0.0	0.0	0.0	1.0	
−1.0	0.0	−2.0	0.0	
3				
0.0	1.0	0.0	0.0	2ND DATA SET
0.0	1.0	−4.0	−6.0	
−4.0				
4				
0.0	1.0	0.0	0.0	3RD DATA SET
0.0	0.0	1.0	0.0	
0.0	0.0	0.0	1.0	
−4.0	−8.0	−8.0	−4.0	
4				
0.0	0.0	1.0	0.0	4TH DATA SET
1.0	0.0	0.0	1.0	
0.0	0.0	0.0	−1.0	
0.0	1.0	0.0	0.0	
	(BLANK CARD)			

Figure 3. Sample input data card set.

1st Data Set Output

```
A MATRIX
   0.                   .1000000000D+01     0.                  0.
   0.                  0.                   .1000000000D+01     0.
   0.                  0.                  0.                   .1000000000D+01
  -.1000000000D+01     0.                  -.2000000000D+01     0.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

CHARACTERISTIC POLYNOMIAL

 (  .10000000D+01) +
 (-0.            ) *X** 1 +
 (  .20000000D+01) *X** 2 +
 (-0.            ) *X** 3 +
 (  .10000000D+01) *X** 4

CHARACTERISTIC ROOTS

REAL PART             COMPLEX PART            MULTIPLICITY
0.                    -.10000000D+01               2
0.                     .10000000D+01               2

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

 EXP(AX) =
 + F( 1)*Z( 1, 1) + F( 2)*ZZ( 1, 1)
 + F( 3)*Z( 1, 2) + F( 4)*ZZ( 1, 2)

 WHERE F(I) ARE
```

Figure 4. Sample program output.

```
F( 1) = COS(X* -.10000000D+01)

F( 2) = SIN(X* -.10000000D+01)

F( 3) = (X** 1)*COS(X* -.10000000D+01)

F( 4) = (X** 1)*SIN(X* -.10000000D+01)

AND WHERE THE Z AND ZZ MATRICES ARE

Z( 1, 1) MATRIX IS

  .1000000000D+01    0.                 0.                 0.
 0.                   .1000000000D+01   0.                 0.
 0.                  0.                  .1000000000D+01   0.
 0.                  0.                 0.                  .1000000000D+01

ZZ( 1, 1) MATRIX IS

 0.                 -.1500000000D+01    0.                 -.500000000D+00
  .500000000D+00     0.                 -.500000000D+00     0.
 0.                   .500000000D+00    0.                 -.500000000D+00
  .500000000D+00     0.                  .1500000000D+01    0.

Z( 1, 2) MATRIX IS

 0.                 -.500000000D+00     0.                 -.500000000D+00
  .500000000D+00     0.                  .500000000D+00     0.
 0.                   .500000000D+00    0.                  .500000000D+00
 -.500000000D+00     0.                 -.500000000D+00     0.

ZZ( 1, 2) MATRIX IS

 -.500000000D+00     0.                 -.500000000D+00     0.
 0.                  -.500000000D+00    0.                 -.500000000D+00
  .500000000D+00     0.                  .500000000D+00     0.
 0.                   .500000000D+00    0.                  .500000000D+00

MAXIMUM ENTRY OF ERROR MATRIX=  .40766752D-12

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Figure 4. (Continued).

2nd Data Set Output

```
A MATRIX
   0.                     .100000000D+01       0.
   0.                     0.                   .100000000D+01
   -.400000000D+01        -.600000000D+01      -.400000000D+01

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

CHARACTERISTIC POLYNOMIAL

(  .40000000D+01) +
(  .60000000D+01) *X** 1 +
(  .40000000D+01) *X** 2 +
(  .10000000D+01) *X** 3

CHARACTERISTIC ROOTS

REAL PART              COMPLEX PART            MULTIPLICITY
-.20000000D+01         0.                           1
-.10000000D+01         -.10000000D+01               1
-.10000000D+01          .10000000D+01               1

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXP(AX) =

+ F( 1)*Z( 1, 1)

+ F( 2)*Z( 2, 1) + F( 3)*ZZ( 2, 1)

WHERE F(I) ARE

F( 1) = EXP(X* -.20000000D+01)
```

Figure 4. (Continued).

```
F( 2) = (X** 0)*EXP(X* -.10000000D+01)*COS(X* -.10000000D+01)


F( 3) = (X** 0)*EXP(X* -.10000000D+01)*SIN(X* -.10000000D+01)


AND WHERE THE Z AND ZZ MATRICES ARE


Z( 1, 1) MATRIX IS

   .100000000D+01        .100000000D+01        .500000000D+00
  -.200000000D+01       -.200000000D+01       -.100000000D+01
   .400000000D+01        .400000000D+01        .200000000D+01


Z( 2, 1) MATRIX IS

  0.                    -.100000000D+01       -.500000000D+00
   .200000000D+01        .300000000D+01        .100000000D+01
  -.400000000D+01       -.400000000D+01       -.100000000D+01


ZZ( 2. 1) MATRIX IS

  -.200000000D+01       -.200000000D+01       -.500000000D+00
   .200000000D+01        .100000000D+01       0.
  0.                     .200000000D+01        .100000000D+01


MAXIMUM ENTRY OF ERROR MATRIX=  .60584518D-27


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Figure 4. (Continued).

3rd Data Set Output

```
A MATRIX
   0.                   .100000000D+01       0.                   0.
   0.                  0.                    .100000000D+01      0.
   0.                  0.                   0.                    .100000000D+01
   -.400000000D+01     -.800000000D+01      -.800000000D+01      -.400000000D+01
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
CHARACTERISTIC POLYNOMIAL

( .40000000D+01) +

( .80000000D+01) *X** 1 +

( .80000000D+01) *X** 2 +

( .40000000D+01) *X** 3 +

( .10000000D+01) *X** 4
```

```
CHARACTERISTIC ROOTS

REAL PART              COMPLEX PART            MULTIPLICITY
-.10000000D+01        -.10000000D+01                2
-.10000000D+01         .10000000D+01                2
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
EXP(AX) =

+ F( 1)*Z( 1, 1) + F( 2)*ZZ( 1, 1)

+ F( 3)*Z( 1, 2) + F( 4)*ZZ( 1, 2)

WHERE F(I) ARE
```

Figure 4. (Continued).

```
F( 1) = (X** 0)*EXP(X* -.10000000D+01)*COS(X* -.10000000D+01)

F( 2) = (X** 0)*EXP(X* -.10000000D+01)*SIN(X* -.10000000D+01)

F( 3) = (X** 1)*EXP(X* -.10000000D+01)*COS(X* -.10000000D+01)

F( 4) = (X** 1)*EXP(X* -.10000000D+01)*SIN(X* -.10000000D+01)

AND WHERE THE Z AND ZZ MATRICES ARE

Z( 1, 1) MATRIX IS

  .1000000000D+01     0.                  0.                  0.
 0.                   .1000000000D+01    0.                  0.
 0.                  0.                   .1000000000D+01    0.
 0.                  0.                  0.                   .1000000000D+01

ZZ( 1, 1) MATRIX IS

 -.2000000000D+01    -.3000000000D+01    -.1500000000D+01    -.5000000000D+00
  .2000000000D+01     .2000000000D+01     .1000000000D+01     .5000000000D+00
 -.2000000000D+01    -.2000000000D+01    -.2000000000D+01    -.1000000000D+01
  .4000000000D+01     .6000000000D+01     .6000000000D+01     .2000000000D+01

Z( 1, 2) MATRIX IS

 -.1000000000D+01    -.2000000000D+01    -.1500000000D+01    -.5000000000D+00
  .2000000000D+01     .3000000000D+01     .2000000000D+01     .5000000000D+00
 -.2000000000D+01    -.2000000000D+01    -.1000000000D+01    0.
 0.                  -.2000000000D+01    -.2000000000D+01    -.1000000000D+01

ZZ( 1, 2) MATRIX IS

 -.1000000000D+01    -.1000000000D+01    -.5000000000D+00    0.
 0.                  -.1000000000D+01    -.1000000000D+01    -.5000000000D+00
  .2000000000D+01     .4000000000D+01     .3000000000D+01     .1000000000D+01
 -.4000000000D+01    -.6000000000D+01    -.4000000000D+01    -.1000000000D+01

MAXIMUM ENTRY OF ERROR MATRIX=  .10020293D-10

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Figure 4. (Continued).

4th Data Set Output

A MATRIX

0.	0.	.1000000000D+01	0.
.1000000000D+01	0.	0.	.1000000000D+01
0.	0.	0.	-.1000000000D+01
0.	.1000000000D+01	0.	0.

* *

CHARACTERISTIC POLYNOMIAL

(.10000000D+01) +

(-0.) *X** 1 +

(-.10000000D+01) *X** 2 +

(-0.) *X** 3 +

(.10000000D+01) *X** 4

CHARACTERISTIC ROOTS

REAL PART	COMPLEX PART	MULTIPLICITY
-.86602540D+00	.50000000D+00	1
-.86602540D+00	-.50000000D+00	1
.86602540D+00	-.50000000D+00	1
.86602540D+00	.50000000D+00	1

* *

EXP(AX) =

+ F(1)*Z(1, 1) + F(2)*ZZ(1, 1)

+ F(3)*Z(3, 1) + F(4)*ZZ(3, 1)

Figure 4. (Continued).

```
WHERE F(I) ARE

F( 1) = (X** 0)*EXP(X* -.866025400D+00)*COS(X*  .500000000D+00)

F( 2) = (X** 0)*EXP(X* -.866025400D+00)*SIN(X*  .500000000D+00)

F( 3) = (X** 0)*EXP(X*  .866025400D+00)*COS(X* -.500000000D+00)

F( 4) = (X** 0)*EXP(X*  .866025400D+00)*SIN(X* -.500000000D+00)

AND WHERE THE Z AND ZZ MATRICES ARE

Z( 1, 1) MATRIX IS

   .5000000000D+00   -.2886751350D+00   -.5773502690D+00    0.
  -.2886751350D+00    .5000000000D+00    0.                -.5773502690D+00
  -.2886751350D+00    0.                 .5000000000D+00    .2886751350D+00
   0.                -.2886751350D+00    .2886751350D+00    .5000000000D+00

ZZ( 1, 1) MATRIX IS

   .2886751350D+00   -.5000000000D+00    0.                 .5773502690D+00
   .5000000000D+00   -.2886751350D+00   -.5773502690D+00    0.
  -.5000000000D+00    .5773502690D+00    .2886751350D+00   -.5000000000D+00
  -.5773502690D+00    .5000000000D+00    .5000000000D+00   -.2886751350D+00

Z( 3, 1) MATRIX IS

   .5000000000D+00    .2886751350D+00    .5773502690D+00    0.
   .2886751350D+00    .5000000000D+00    0.                 .5773502690D+00
   .2886751350D+00    0.                 .5000000000D+00   -.2886751350D+00
   0.                 .2886751350D+00   -.2886751350D+00    .5000000000D+00

ZZ( 3, 1) MATRIX IS

   .2886751350D+00    .5000000000D+00    0.                 .5773502690D+00
  -.5000000000D+00   -.2886751350D+00   -.5773502690D+00    0.
   .5000000000D+00    .5773502690D+00    .2886751350D+00    .5000000000D+00
  -.5773502690D+00   -.5000000000D+00   -.5000000000D+00   -.2886751350D+00

MAXIMUM ENTRY OF ERROR MATRIX=  .378653230-28
```

Figure 4. (Concluded).

```
      PROGRAM EAT(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      DIMENSION XCOF(10),COF(10),ROOTR(10),ROOTI(10)
      DIMENSION STORX(10),STORY(10)
      DIMENSION VR(10,10),VC(10,10),RSAV(10,10)
      DIMENSION KF(10),N(10),RELL(10),CMPXX(10)
      DIMENSION A(10,10),Z(10,10),ZZ(10,10),PROD(20,20)
      DIMENSION V(20,20),S(400)
      DIMENSION L1(20),M1(20)
      DOUBLE PRECISION RELL,CMPXX,A,Z,ZZ,PROD
      DOUBLE PRECISION V,S,O,B
      DOUBLE PRECISION DA,DB,DC,DD
      DOUBLE PRECISION XCOF,COF,ROOTR,ROOTI,STORY,STORX
      DOUBLE PRECISION Y,X
      DOUBLE PRECISION YY
      DOUBLE PRECISION VR,VC,RSAV
C
C     READ IN ALL THE DATA
C
      READ(5,101) DB
      READ(5,101) DC
      READ(5,101) DD
  101 FORMAT(1D15.0)
  866 WRITE(6,755)
      WRITE(6,937)
  937 FORMAT(60(1X,1H*))
      WRITE(6,801)
 801  FORMAT(1H1)
      READ(5,1) LDEM
      K=0
      IFF=0
      IF(LDEM) 761,762,761
  761 READ(5,88)((A(I,J),J=1,LDEM),I=1,LDEM)
   88 FORMAT(4D15.0)
      WRITE(6,60)
   60 FORMAT(//////3X,8HA MATRIX)
      CALL WRITE1(LDEM,A)
      WRITE(6,755)
      WRITE(6,937)
C
C     CALCULATE THE CHARACTERISTIC POLYNOMIAL
C
      XCOF(LDEM+1)=1.D0
      CALL TRACE(A,LDEM,B)
      XCOF(LDEM)=-B
      DO 40 I=1,LDEM
      DO 40 J=1,LDEM
   40 VR(I,J)=A(I,J)
      DO 10 I=1,LDEM
   10 VR(I,I)=VR(I,I)+XCOF(LDEM)
      MM=LDEM-1
      M=LDEM
      DO 33 I=1,MM
      CALL MATPRO(M,A,VR,PROD)
      CALL TRACE2(PROD,M,B)
      XCOF(M-I)=-(1.D0/DBLE(FLOAT(I+1)))*B
      DO 20 IP=1,M
      DO 20 JP=1,M
```

Figure 5. Program listing.

```
   20 VR(IP,JP)=PROD(IP,JP)
      DO 30 IP=I,M
   30 VR(IP,IP)=VR(IP,IP)+XCOF(M-I)
   33 CONTINUE
      WRITE(6,755)
      WRITE(6,104)
  104 FORMAT(3X,25HCHARACTERISTIC POLYNOMIAL//)
      CALL WRITEP(XCOF,LOEM)
C
C     CALCULATE THE CHARACTERISTIC ROOTS
C
      CALL POLRT(XCOF,COF,LOEM,ROOTR,ROOTI,IER)
      DO 399 I=1,LOEM
      IF(ROOTR(I).LT.DB.AND.ROOTR(I).GT.-DB) ROOTR(I)=0.D0
      IF(ROOTI(I).LT.DB.AND.ROOTI(I).GT.-DB) ROOTI(I)=0.D0
  399 CONTINUE
      IF(IER) 6I,70,61
   6I WRITE(6,I08) IER
  108 FORMAT(///3X,30HERROR IN ROOT CALCULATION,MODE,I2)
      GO TO 866
C
C     SORT AND CLASSIFY THE ROOTS
C
   70 CALL SORT(ROOTR,ROOTI,RELL,CMPXX,STORX,STORY,N,LOEM,K,DB)
      WRITE(6,755)
  755 FORMAT(3X///)
      WRITE(6,55)
   55 FORMAT(2X,20HCHARACTERISTIC ROOTS  //2X,9HREAL PART,13X,12HCOMPLEX
     1 PART,13X,12HMULTIPLICITY    )
      IQ=K
      DO 700 I=I,IQ
  700 WRITE(6,701) RELL(I) ,CMPXX(I),N(I)
  701 FORMAT(IX,D15.8,5X,D15.8,I5X,I5)
      WRITE(6,755)
      WRITE(6,937)
C
C     CALCULATE AND INVERT THE VANDERMONDE
C
      KF(I)=I
      DO 883 I=I,LOEM
      KFX=I+I
  883 KF(I+I)=KFX*KF(I)
      DO 3I3 III=I,K
      IF(CMPXX(III).NE.0.D0) GO TO I29
  3I3 CONTINUE
      CALL VANDR(RELL,LOEM,N,K,V)
      GO TO 107
  I29 LOEM2=2*LOEM
      CALL VAND(RELL,CMPXX,LOEM,N,K,V)
      CALL ARRAY(2,LOEM2,LOEM2,20,20,S,V)
      CALL GINVRT(S,LOEM2,D,LI,MI)
      CALL ARRAY(I,LOEM2,LOEM2,20,20,S,V)
      D=DABS(D)
      D=DSQRT(D)
      IF(D.GT.DD.OR.D.LT.-DD) GO TO 646
  647 WRITE(6,648)
  648 FORMAT(3X,40HV MATRIX IS SINGULAR -- CHECK INPUT DATA)
```

Figure 5. (Continued).

```
              GO TO 866
          646 KFG=1
              GO TO 555
        107   CALL ARRAY(2,LOEM,LDEM,20,20,S,V)
              CALL GINVRT(S,LOEM,D,LI,MI)
              CALL ARRAY(1,LDEM,LDEM,20,20,S,V)
              IF(D.LT.DD.AND.D.GT.-DD) GO TO 647
        C
        C     DETERMINE THE FORM OF EXP(AT)
        C
          649 KFG=0
              LOEM2=LDEM
          555 IK=0
              WRITE(6,756)
          756 FORMAT(3X///)
              WRITE(6,411)
              IIK=0
              KOOP=0
              DO 444 I=I,K
              IF(KOOP) 414,415,414
          414 KOOP=0
              GO TO 444
          415 LGM=N(I)
              DO 400 J=I,LGM
              IIK=IIK+1
              IF(CMPXX(I)) 406,401,406
          401 WRITE(6,410) IIK,I,J
              GO TO 400
          406 LOG=IIK+1
              WRITE(6,440) IIK,I,J,LOG,I,J
              IIK=LOG
              KOOP=1
          400 CONTINUE
          444 CONTINUE
          411 FORMAT(3X,9HEXP(AX) = /)
          440 FORMAT(3X,4H+ F(,I2,4H)*Z(,I2,1H,,I2,6H) + F(,I2,5H)*ZZ(,I2,1H,,
             1I2,1H)/)
          410 FORMAT(3X,4H+ F(,I2,4H)*Z(,I2,1H,,I2,1H)/)
              WRITE(6,430)
              KOOP=0
              IIK=0
              DO 901 I=1,K
              LGM=N(I)
              IF(KOOP) 614,615,614
          614 KOOP=0
              GO TO 901
          615 DO 600 J=I,LGM
              IIK=IIK+1
              IF(CMPXX(I)) 606,601,606
          601 IF(J-1) 604,602,604
          602 WRITF(6,603) IIK,RELL(I)
              GO TO 600
          604 JJJ=J-1
              WRITE(6,605) IIK,JJJ,RELL(I)
              GO TO 600
          606 KOOP=1
              IF(RELL(I)) 612,607,612
```

Figure 5. (Continued).

```
      607 IF(J-1) 610,608,610
      608 WRITE(6,609) IIK,CMPXX(I)
          IIK=IIK+1
          WRITE(6,709) IIK,CMPXX(I)
          KOOP=1
          GO TO 600
      610 JJJ=J-1
          WRITE(6,611) IIK,JJJ,CMPXX(I)
          IIK=IIK+1
          WRITE(6,711) IIK,JJJ,CMPXX(I)
          GO TO 600
      612 JJJ=J-1
          WRITE(6,613) IIK,JJJ,RELL(I),CMPXX(I)
          IIK=IIK+1
          WRITE(6,713) IIK,JJJ,RELL(I),CMPXX(I)
      600 CONTINUE
      901 CONTINUE
      605 FORMAT(///3X,2HF(,I2,8H) = (X**,I2,8H)*EXP(X*,D15.8,1H))
      611 FORMAT(///3X,2HF(,I2,8H) = (X**,I2,8H)*COS(X*,D15.8,1H))
      711 FORMAT(///3X,2HF(,I2,8H) = (X**,I2,8H)*SIN(X*,D15.8,1H))
      613 FORMAT(///3X,2HF(,I2,8H) = (X**,I2,8H)*EXP(X*,D15.8,
         18H)*COS(X*,D15.8,1H))
      713 FORMAT(///3X,2HF(,I2,8H) = (X**,I2,8H)*EXP(X*,D15.8,
         18H)*SIN(X*,D15.8,1H))
      603 FORMAT(///3X,2HF(,I2,10H) = EXP(X*,D15.8,1H))
      609 FORMAT(///3X,2HF(,I2,4H) = ,6HCOS(X*,D15.8,1H))
      709 FORMAT(///3X,2HF(,I2,4H) = ,6HSIN(X*,D15.8,1H))
      430 FORMAT(///3X,14HWHERE F(I) ARE)
          WRITE(6,760)
  760     FORMAT(///3X,35HAND WHERE THE Z AND ZZ MATRICES ARE    )
C
C         CALCULATE THE CONSTITUENT MATRICES
C
          KOOP=0
          IK=0
          YY=0.D0
          DO 809 I=1,K
          IF(KOOP) 616,617,616
      616 KOOP=0
          IK=IK+1
          GO TO 809
      617 MMM=N(I)
          DO 109 J=1,MMM
          IK=IK+1
          IF(CMPXX(I))618,619,618
      618 KOOP=1
          KFG=1
          GO TO 620
      619 KFG=0
      620 DO 111 LIL=1,LDEM
          DO 111 LJL=1,LDEM
          IF(LIL-LJL) 113,112,113
      112 Z(LIL,LJL)=V(IK,1)
          GO TO 111
      113 Z(LIL,LJL)=0.0
      111 CONTINUE
          IF(KFG) 205,206,205
```

Figure 5. (Continued).

```
      205 DO 114 ILL=1,LDEM
          DO 114 JLL=1,LDEM
          IF(ILL-JLL)116,115,116
      115 MKM=IK+LDEM
          ZZ(ILL,JLL)=V(MKM,1)
          GO TO 114
      116 ZZ(ILL,JLL) =0.0
      114 CONTINUE
      206 DO 140 JL=2,LDEM
          IF(JL.EQ.2) GO TO 173
          IF(JL.EQ.3) GO TO 132
          CALL MATPRO(LDEM,A,RSAV,PROD)
          GO TO 273
      132 DO 236 IP=1,LDEM
          DO 236 JP=1,LDEM
      236 RSAV(IP,JP)=A(IP,JP)
          CALL MATPRO(LDEM,A,RSAV,PROD)
      273 DO 199 IP=1,LDEM
          DO 199 JP=1,LDEM
      199 RSAV(IP,JP)=PROD(IP,JP)
          GO TO 120
      173 DO 919 IP=1,LDEM
          DO 919 JP=1,LDEM
      919 PROD(IP,JP)=A(IP,JP)
      120 DO 119 IN=1,LDEM
          DO 119 JN=1,LDEM
          Z(IN,JN)=Z(IN,JN)+V(IK,JL)*PROD(IN,JN)
          IF(KFG) 333,119,333
      333 LKL=IK+LDEM
          ZZ(IN,JN)=ZZ(IN,JN)+V(LKL,JL)*PROD(IN,JN)
      119 CONTINUE
      140 CONTINUE
          IF(KFG) 758,757,758
      758 DO 759 MIN=1,LDEM
          DO 759 MJN=1,LDEM
          Z(MIN,MJN)=2.D0*Z(MIN,MJN)
      759 ZZ(MIN,MJN)=-2.D0*ZZ(MIN,MJN)
      757 WRITE(6,778) I,J
          IF(J.EQ.1) X=1.D0
          IF(J.GT.1) X=DBLE(FLOAT(KF(J-1)))
          DO 827 MU=1,LDEM
          DO 827 MUT=1,LDEM
          Z(MU,MUT)=Z(MU,MUT)/X
      827 CONTINUE
          DO 321 MU=1,LDEM
          DO 321 MUT=1,LDEM
          X=Z(MU,MUT)
          IF(X.LT.DC.AND.X.GT.-DC) Z(MU,MUT)=0.D0
      321 CONTINUE
          CALL WRITE1 (LDEM,Z)
          IF(KFG) 207,309,207
      309 X=RELL(I)
          IF(J.EQ.1) GO TO 157
          CALL MATPRO(LDEM,A,VR,PROD)
          DO 837 IP=1,LDEM
          DO 837 JP=1,LDEM
          Y=X*VR(IP,JP)+Z(IP,JP)*FLOAT(J-1)-PROD(IP,JP)
```

Figure 5. (Continued).

```
      Y=DABS(Y)
   14 IF(Y-YY) 837,17,17
   17 YY=Y
  837 CONTINUE
  157 DO 783 IP=1,LOEM
      DO 783 JP=1,LOEM
  783 VR(IP,JP)=Z(IP,JP)
      IF(J.NE.MMM) GO TO 109
      CALL MATPRO(LOEM,A,VR,PROD)
      DO 387 IP=1,LOEM
      DO 387 JP=1,LOEM
      Y=PROD(IP,JP)-VR(IP,JP)*X
      Y=DABS(Y)
   15 IF(Y-YY) 387,16,16
   16 YY=Y
  387 CONTINUE
  856 IF(KFG) 207,109,207
  207 WRITE(6,779) I,J
      IF(J.EQ.1) X=1.00
      IF(J.GT.1) X=DBLE(FLOAT(KF(J-1)))
      DO 839 MU=1,LOEM
      DO 839 MUT=1,LOEM
      ZZ(MU,MUT)=ZZ(MU,MUT)/X
  839 CONTINUE
      DO 128 MU=1,LOEM
      DO 128 MUT=1,LOEM
      X=ZZ(MU,MUT)
      IF(X.LT.DC.AND.X.GT.-DC) ZZ(MU,MUT)=0.00
  128 CONTINUE
      CALL WRITE1(LOEM,ZZ)
      IF(J.EQ.1) GO TO 717
      CALL MATPRO(LOEM,A,VR,PROD)
      DO 703 IP=1,LOEM
      DO 703 JP=1,LOEM
      Y=FLOAT(J-1)*Z(IP,JP)+RELL(I)*VR(IP,JP)+CMPXX(I)*VC(IP,JP)
      Y=Y-PROD(IP,JP)
      Y=DABS(Y)
   21 IF(Y-YY) 703,22,22
   22 YY=Y
  703 CONTINUE
      CALL MATPRO(LOEM,A,VC,PROD)
      DO 370 IP=1,LOEM
      DO 370 JP=1,LOEM
      Y=FLOAT(J-1)*ZZ(IP,JP)+RELL(I)*VC(IP,JP)-CMPXX(I)*VR(IP,JP)
      Y=Y-PROD(IP,JP)
      Y=DABS(Y)
   23 IF(Y-YY) 370,24,24
   24 YY=Y
  370 CONTINUE
  717 DO 307 IP=1,LOEM
      DO 307 JP=1,LOEM
      VR(IP,JP)=Z(IP,JP)
      VC(IP,JP)=ZZ(IP,JP)
  307 CONTINUE
      IF(J.NE.MMM) GO TO 109
      CALL MATPRO(LOEM,A,VR,PROD)
      DO 893 IP=1,LOEM
      DO 893 JP=1,LOEM
      Y=RELL(I)*VR(IP,JP)+CMPXX(I)*VC(IP,JP)-PROD(IP,JP)
      Y=DABS(Y)
   41 IF(Y-YY) 893,42,42
   42 YY=Y
  893 CONTINUE
      CALL MATPRO(LOEM,A,VC,PROD)
      DO 993 IP=1,LOEM
      DO 993 JP=1,LOEM
      Y=CMPXX(I)*VR(IP,JP)-RELL(I)*VC(IP,JP)+PROD(IP,JP)
      Y=DABS(Y)
   43 IF(Y-YY) 993,44,44
   44 YY=Y
  993 CONTINUE
  109 CONTINUE
  889 CONTINUE
      WRITE(6,755)
      WRITE(6,19) YY
   19 FORMAT(3X,30HMAXIMUM ENTRY OF ERROR MATRIX=,1D15.8)
      GO TO 866
  779 FORMAT(///3X,3HZZ(,I2,1H,,I2,11H) MATRIX IS/)
  778 FORMAT(///3X,2HZ(,I2,1H,,I2,11H) MATRIX IS/)
    1 FORMAT(I2)
  762 CONTINUE
      END
```

Figure 5. (Continued).

```
SUBROUTINE GINVRT      74/74    OPT=1                    FTN 4.2+75067       05/16/75  09.59.53.

            SUBROUTINE GINVRT(A,N,D,L,M)
      C
      C     THIS ROUTINE CALCULATES THE INVERSE OF A MATRIX
      C
            DIMENSION A(1),L(1),M(1)
            DOUBLE PRECISION A,BIGA,D,HOLD
            D=1.
            NK=-N
            DO 80 K=1,N
            NK=NK+N
            L(K)=K
            M(K)=K
            KK=NK+K
            BIGA=A(KK)
            DO 20 J=K,N
            IZ=N*(J-1)
            DO 20 I=K,N
            IJ=IZ+I
       10   IF(DABS(BIGA)-DABS(A(IJ))) 15,20,20
       15   BIGA=A(IJ)
            L(K)=I
            M(K)=J
       20   CONTINUE
      C     INTERCHANGE ROWS
            J=L(K)
            IF(J-K)  35,35,25
       25   KI=K-N
            DO 30  I=1,N
            KI=KI+N
            HOLD=-A(KI)
            JI=KI-K+J
            A(KI)=A(JI)
       30   A(JI)=HOLD
      C     INTERCHANGE COLUMNS
       35   I=M(K)
            IF(I-K) 45,45,38
       38   JP=N*(I-1)
            DO 40 J=1,N
            JK=NK+J
            JI=JP+J
            HOLD=-A(JK)
            A(JK)=A(JI)
       40   A(JI)=HOLD
      C     DIVIDE COLUMN BY MINUS PIVOT  (VALUE OF PIVOT ELEMENT IS CONTAINED
      C     IN BIGA)
       45   IF(BIGA)  48,46,48
       46   D=0.
            RETURN
       48   DO 55 I=1,N
            IF(I-K) 50,55,50
       50   IK=NK+I
            A(IK)=A(IK)/(-BIGA)
       55   CONTINUE
      C     REDUCE MATRIX
            DO 65 I=1,N
            IK=NK+I
            HOLD=A(IK)
```

Figure 5. (Continued).

```
      IJ=I-N
      DO 65 J=1,N
      IJ=IJ+N
      IF(I-K) 60,65,60
   60 IF(J-K) 62,65,62
   62 KJ=IJ-I+K
      A(IJ)=HOLD*A(KJ)+A(IJ)
   65 CONTINUE
C     DIVIDE ROW BY PIVOT
      KJ=K-N
      DO 75 J=1,N
      KJ=KJ+N
      IF(J-K) 70,75,70
   70 A(KJ)=A(KJ)/BIGA
   75 CONTINUE
C     PRODUCTA OF PIVOTS
      D=D*BIGA
C     REPLACE PIVOT BY RECIPROCAL
      A(KK)=1./BIGA
   80 CONTINUE
C     FINAL ROW AND COLUMN INTERCHANGE
      K=N
  100 K=(K-1)
      IF(K) 150,150,105
  105 I=L(K)
      IF(I-K) 120,120,108
  108 JQ=N*(K-1)
      JR=N*(I-1)
      DO 110 J=1,N
      JK=JQ+J
      HOLD=A(JK)
      JI=JR+J
      A(JK)=-A(JI)
  110 A(JI)=HOLD
  120 J=M(K)
      IF(J-K) 100,100,125
  125 KI=K-N
      DO 130 I=1,N
      KI=KI+N
      HOLD=A(KI)
      JI=KI-K+J
      A(KI)=-A(JI)
  130 A(JI)=HOLD
      GO TO 100
  150 RETURN
      END
```

Figure 5. (Continued).

```
SUBROUTINE ARRAY       74/74    OPT=1                          FTN 4.2+75067       05/16/75  09.59.58.

                SUBROUTINE ARRAY (MODE,I,J,M,N,S,D)
      C
      C         THIS ROUTINE PREPARES A MATRIX FOR SINVERT
      C
                DIMENSION S(1),D(1)
                DOUBLE PRECISION S,D
                NI=N-I
      C         TEST TYPE OF CONVERSION
                IF(MODE-1)  100,100,120
      C         CONVERT FROM SINGLE TO DOUBLE DIMENSION
      100       IJ=I*J+1
                NM=N*J+1
                DO 110  K=1,J
                NM=NM-NI
                DO 110 L=1,I
                IJ=IJ-1
                NM=NM-1
      110       D(NM)=S(IJ)
                GO TO 140
      C         CONVERT FROM DOUBLE TO SINGLE
      120       IJ=0
                NM=0
                DO 130 K=1,J
                DO 125 L=1,I
                IJ=IJ+1
                NM=NM+1
      125       S(IJ)=D(NM)
      130       NM=NM+NI
      140       RETURN
                END
```

```
SUBROUTINE MATPRO      74/74    OPT=1                          FTN 4.2+75067       05/16/75  10.00.00.

                SUBROUTINE MATPRO(LDEM,A,B,PROD)
      C
      C         THIS ROUTINE CALCULATES THE PRODUCT OF TWO MATRICES
      C
                DIMENSION A(10,10),B(10,10),PROD(20,20)
                DOUBLE PRECISION A,B, PROD
                DO 20 I=1,LDEM
                DO 20 J=1,LDEM
                PROD(I,J)=0.
                DO 20 L=1,LDEM
      20        PROD(I,J)=PROD(I,J)+A(I,L)*B(L,J)
                RETURN
                END
```

```
SUBROUTINE WRITE1      74/74    OPT=1                          FTN 4.2+75067       05/16/75  10.01.12.

                SUBROUTINE WRITE1 (IQ,A)
      C
      C         THIS ROUTINE WRITES A MATRIX
      C
                DIMENSION A(10,10)
                DOUBLE PRECISION A
                DO 1 I=1,IQ
      1         WRITE(6,2) (A(I,J),J=1,IQ)
      2         FORMAT(6(4X,D16.9))
                RETURN
                END
```

Figure 5. (Continued).

```
SUBROUTINE VAND      74/74    OPT=1                      FTN 4.2+75067      05/16/75  10.01.14.

      SUBROUTINE VAND(REALP,COMPP,N,M,K,V)
C
C     THIS ROUTINE CALCULATES THE VANDERMONDE FOR MATRICES WITH COMPLEX
C     EIGENVALUES
C
      DIMENSION REALP(1),COMPP(1),M(1),V(20,20)
      DOUBLE PRECISION V,REALP,COMPP
      DOUBLE PRECISION A,B,C,D,E,F,AA,BB
      NSUM=0
      DO 10 L=1,K
      MM=M(L)
      IF(L.EQ.1) NC=1
      IF(L.GT.1) NC=NSUM+1
      NSUM=NSUM+MM
      A=REALP(L)
      B=COMPP(L)
      C=1.D0
      D=0.D0
      DO 30 I=2,N
      CALL COMCAL(A,B,C,D,E,F)
      C=E
      D=F
      V(I+N,NC)=D
      V(I,NC+N)=-D
      V(I+N,NC+N)=V(I,NC)=C
   30 CONTINUE
      V(1+N,NC+N)=V(1,NC)=1.D0
      V(1+N,NC)=V(1,NC+N)=0.D0
      IF(MM.EQ.1) GO TO 10
      DO 40 I=1,N
      DO 40 J=2,MM
      JJ=NC+J-1
      IF(I-J) 97,98,99
   97 V(I+N,JJ+N)=V(I,JJ)=0.D0
      V(I+N,JJ)=V(I,JJ+N)=0.D0
      GO TO 40
   98 V(I+N,JJ+N)=V(I,JJ)=1.D0
      V(I+N,JJ)=V(I,JJ+N)=0.D0
      GO TO 40
   99 AA=V(I-1,JJ)
      BB=V(I+N-1,JJ)
      C=V(I-1,JJ-1)
      D=V(I+N-1,JJ-1)
      CALL COMCAL(AA,BB,A,B,E,F)
      E=E+C
      F=F+D
      V(I+N,JJ+N)=V(I,JJ)=E
      V(I+N,JJ)=F
      V(I,N+JJ)=-F
   40 CONTINUE
   10 CONTINUE
      RETURN
      END
```

Figure 5. (Continued).

```
SUBROUTINE COMCAL      74/74    OPT=1                      FTN 4.2+75067      05/16/75  10.02.01.

              SUBROUTINE COMCAL(A,B,C,D,E,F)
        C
        C     THIS ROUTINE CALCULATES THE PRODUCT OF A DOUBLE PRECISION COMPLEX
        C     NUMBER
        C
              DOUBLE PRECISION A,B,C,D,E,F
              E=A*C-B*D
              F=A*D+B*C
              RETURN
              END
```

```
SUBROUTINE VANDR       74/74    OPT=1                      FTN 4.2+75067      05/16/75  10.02.53.

              SUBROUTINE VANDR(RELL,N,M,K,V)
        C
        C     THIS ROUTINE CALCULATES THE VANDERMONDE FOR A MATRIX WITH ONLY REAL
        C     EIGENVALUES
        C
              DIMENSION RELL(1),M(1),V(20,20)
              DOUBLE PRECISION RELL,V
              DOUBLE PRECISION Z,ZZ
              NSUM=0
              DO 10 L=1,K
              MM=M(L)
              IF(L.EQ.1) NC=1
              IF(L.GT.1) NC=NSUM+1
              NSUM=NSUM+MM
              ZZ=1.D0
              Z=RELL(L)
              DO 30 I=2,N
              ZZ=ZZ*Z
              V(I,NC)=ZZ
           30 CONTINUE
              V(1,NC)=1.D0
              IF(MM.EQ.1) GO TO 10
              DO 40 I=1,N
              DO 40 J=2,MM
              JJ=NC+J-1
              IF(I-J) 97,98,99
           97 V(I,JJ)=0.D0
              GO TO 40
           98 V(I,JJ)=1.D0
              GO TO 40
           99 V(I,JJ)=V(I-1,JJ-1)+V(I-1,JJ)*Z
           40 CONTINUE
           10 CONTINUE
              RETURN
              END
```

```
SUBROUTINE WRITEP      74/74    OPT=1                      FTN 4.2+75067      05/16/75  10.02.58.

              SUBROUTINE WRITEP(XCOF,N)
        C
        C     THIS ROUTINE WRITES A POLYNOMIAL
        C
              DIMENSION XCOF(1)
              DOUBLE PRECISION XCOF
              WRITE(6,103) XCOF(1)
              NN=N-1
              DO 10 I=1,NN
           10 WRITE(6,100) XCOF(I+1),I
              WRITE(6,101) XCOF(N+1),N
          100 FORMAT(/3X,1H(,1D15.8,1H),1X,4H*X**,12,1X,1H+)
          101 FORMAT(/3X,1H(,1D15.8,1H),1X,4H*X**,12)
          103 FORMAT(/3X,1H(,1D15.8,1H),1X,1H+)
              RETURN
              END
```

Figure 5. (Continued).

```
      SUBROUTINE SORT(ROOTR,ROOTI,RELL,CMPXX,STORX,STORY,M,N,K,DK)
C
C     THIS ROUTINE SORTS THE ROOTS
C
      DOUBLE PRECISION DK
      DIMENSION M(1)
      DIMENSION STORX(1),STORY(1)
      DIMENSION ROOTR(1),ROOTI(1),RELL(1),CMPXX(1)
      DOUBLE PRECISION ROOTR,ROOTI,RELL,CMPXX
      DOUBLE PRECISION STORX,STORY
      DOUBLE PRECISION X,Y,W,Z,DD,DB
      K=0
      KK=0
      DO 10 I=1,N
      K=K+1
      M(K)=0
      X=ROOTR(1)
      Y=ROOTI(1)
      RELL(K)=X
      CMPXX(K)=Y
      NN=N-KK
      NK=0
      DO 20 J=1,NN
      W=ROOTR(J)
      Z=ROOTI(J)
      DD=DABS(X-W)
      DB=DABS(Y-Z)
      IF(DD.LT.DK.AND.DB.LT.DK) GO TO 30
      NK=NK+1
      STORX(NK)=W
      STORY(NK)=Z
      GO TO 20
   30 M(K)=M(K)+1
   20 CONTINUE
      DO 40 L=1,NK
      ROOTR(L)=STORX(L)
      ROOTI(L)=STORY(L)
   40 CONTINUE
      KK=KK+M(K)
      IF(KK.EQ.N) GO TO 50
   10 CONTINUE
   50 RETURN
      END
```

Figure 5. (Continued).

```
      SUBROUTINE POLRT(XCOF,COF,M,ROOTR,ROOTI,IER)
C
C     THIS ROUTINE CALCULATES THE ROOTS OF A POLYNOMIAL
C
      DOUBLE PRECISION FI
      DOUBLE PRECISION XCOF,COF,ROOTR,ROOTI
      DOUBLE PRECISION XO,YO,X,Y,XPR,YPR,UX,UY,V,YT,XT,U,XT2,YT2,SUMSQ
      DOUBLE PRECISION OX,OY,TEMP,ALPHA
      DIMENSION XCOF(1),COF(1),ROOTR(1),ROOTI(1)
      IFIT=0
      N=M
      IER=0
      IF(XCOF(N+1)) 10,25,10
   10 IF(N) 15,15,32
   15 IER=1
   20 RETURN
   25 IER=4
      GO TO 20
   30 IER=2
      GO TO 20
   32 IF(N-36) 35,35,30
   35 NX=N
      NXX=N+1
      N2=1
      KJI=N+1
      DO 40 L=1,KJI
      MT=KJI-L+1
   40 COF(MT)=XCOF(L)
   45 XO=.00500101
      YO=.01000101
      IN=0
   50 X=XO
      XO=-10.0*YO
      YO=-10.0*X
      X=XO
      Y=YO
      IN=IN+1
      GO TO 59
   55 IFIT=1
      XPR=X
      YPR=Y
   59 ICT=0
   60 UX=0.0
      UY=0.0
      V=0.0
      YT=0.0
      XT=1.0
      U=COF(N+1)
      IF(U) 65,130,65
   65 DO 70 I=1,N
      L=N-I+1
      TEMP=COF(L)
      XT2=X*XT-Y*YT
      YT2=X*YT+Y*XT
      U=U+TEMP*XT2
      V=V+TEMP*YT2
      FI=I
```

Figure 5. (Continued).

```
SUBROUTINE POLRT      74/74    OPT=1                        FTN 4.2+75067      05/16/75  10.08.[illegible]

                   UX=UX+FI*XT*TEMP
                   UY=UY-FI*YT*TEMP
                   XT=XT2
                70 YT=YT2
                   SUMSQ=UX*UX+UY*UY
                   IF (SUMSQ) 75,110,75
                75 DX=(V*UY-U*UX) /SUMSQ
                   X=X+DX
                   DY=-(U*UY+V*UX)/SUMSQ
                   Y=Y+DY
                78 IF(DABS(DY)+DABS(DX)-1.D-12) 100,80,80
                80 ICT=ICT+1
                   IF(ICT-500) 60,85,85
                85 IF(IFIT) 100,90,100
                90 IF(IN-5) 50,95,95
                95 IER=3
                   GO TO 20
               100 DO 105 L=1,NXX
                   MT=KJ1-L+1
                   TEMP=XCOF(MT)
                   XCOF(MT)=COF(L)
               105 COF(L)=TEMP
                   ITEMP=N
                   N=NX
                   NX=ITEMP
                   IF(IFIT) 120,55,120
               110 IF(IFIT) 115,50,115
               115 X=XPR
                   Y=YPR
               120 IFIT=0
               122 IF(DABS(Y)-1.D-10*DABS(X)) 135,125,125
               125 ALPHA=X+X
                   SUMSQ=X*X+Y*Y
                   N=N-2
                   GO TO 140
               130 X=0.0
                   NX=NX-1
                   NXX=NXX-1
               135 Y=0.0
                   SUMSQ=0.0
                   ALPHA=X
                   N=N-1
               140 COF(2)=COF(2)+ALPHA*COF(1)
               145 DO 150 L=2,N
               150 COF(L+1)=COF(L+1)+ALPHA*COF(L)-SUMSQ*COF(L-1)
               155 ROOTI(N2)=Y
                   ROOTR(N2)=X
                   N2=N2+1
                   IF(SUMSQ) 160,165,160
               160 Y=-Y
                   SUMSQ=0.0
                   GO TO 155
               165 IF(N) 20,20,45
                   END
```

```
SUBROUTINE TRACE      74/74    OPT=1                        FTN 4.2+75067      05/16/75  10.10.24.

                   SUBROUTINE TRACE(A,N,TR)
             C
             C     THIS ROUTINE CALCULATES THE TRACE OF A MATRIX
             C
                   DIMENSION A(10,10)
                   DOUBLE PRECISION SUM,A,TR
                   SUM=0.D0
                   DO 10 I=1,N
                10 SUM=SUM+A(I,I)
                   TR=SUM
                   RETURN
                   END
```

```
SUBROUTINE TRACE2     74/74    OPT=1                        FTN 4.2+75067      05/16/75  10.10.29.

                   SUBROUTINE TRACE2(A,N,TR)
             C
             C     THIS ROUTINE CALCULATES THE TRACE OF A MATRIX
             C
                   DIMENSION A(20,20)
                   DOUBLE PRECISION SUM,A,TR
                   SUM=0.D0
                   DO 10 I=1,N
                10 SUM=SUM+A(I,I)
                   TR=SUM
                   RETURN
                   END
```

Figure 5. (Concluded).